On the Role of Structured Pruning for Neural Network Compression

ON THE ROLE OF STRUCTURED PRUNING FOR NEURAL NETWORK COMPRESSION

Andrea Bragagnolo[⋆†] *Enzo Tartaglione*[⋆] *Attilio Fiandrotti*[⋆] *Marco Grangetto*[⋆]

[⋆] Computer Science Department, University of Turin, 10149 Torino, Italy
[†]Synesthesia s.r.l., Italy

ABSTRACT

This works explores the benefits of structured parameter pruning in the framework of the MPEG standardization efforts for neural network compression. First less relevant parameters are pruned from the network, then remaining parameters are quantized and finally quantized parameters are entropy coded. We consider an unstructured pruning strategy that maximizes the number of pruned parameters at the price of randomly sparse tensors and a structured strategy that prunes fewer parameters yet yields regularly sparse tensors. We show that structured pruning enables better end-to-end compression despite lower pruning ratio because it boosts the efficiency of the arithmetic coder. As a bonus, once decompressed, the network memory footprint is lower as well as its inference time.

***Index Terms*—** Pruning, Deep learning, Compression, MPEG-7

1. INTRODUCTION

Deep neural networks achieve top-notch performance in a number of challenging multimedia-related tasks thanks to topologies with millions of learnable parameters. For example, the parameters count of popular architectures such as AlexNet, VGG or ResNet is in the order of the tens or hundreds of millions. These numbers pose serious challenges in a number of practical scenarios. For example, large architectures may just not fit in embedded devices such as smartphones where the storage and run-time memory are limited. In federated learning scenarios, where the nodes must frequently exchange the learned parameters, the achievable learning rate may be bounded by the available bandwidth between nodes [1].

The need for compressing deep neural networks (DNNs) prompted the Moving Pictures Experts Group (MPEG) of the International Organization for Standardization (ISO) to define the upcoming MPEG-7 Part 17 standard *Compression of neural networks for multimedia content description and analysis* [2]. The MPEG neural network compression pipeline includes three stages. First, the number of parameters in the neural network is preliminary reduced, e.g. pruning away disposable parameters and yielding a sparse network topology. Second, the parameters that survived pruning are quantized over a finite set of values. Third, the quantized parameters are entropy coded with context adaptive arithmetic coding [3], producing a compressed bitstream. Experiments show that favorable trade-off between compression efficiency and performance could be stricken [4].

While quantization and entropy coding has received more attention, the role of the parameter pruning has not been fully explored. *Unstructured* pruning approaches maximize the number of pruned parameters (for a given performance target) without constraints on the resulting tensor topology [5, 6, 7, 8, 9, 10, 11]. These approaches usually deliver top pruning ratio, i.e. fraction of pruned parameters, at the price of randomly sparse parameter tensor. *Structured* approaches rely on some constraints on the pruning to impose a somewhat regular structure over the pruned topology [12, 13, 14]. These approaches may yield network representations that are easier to hold into memory, i.e. lower run-time memory footprint, despite the pruning ratio is usually lower in reason of the additional constraint. However, the effect of the pruned scheme and the resulting tensors structure on the efficiency of a complete neural network compression pipeline is not totally understood.

In this work we investigate the benefits of structured pruning approaches within MPEG neural network compression pipeline. We describe two pruning strategies that both prune parameters that are less relevant to the network performance: the first approach just aims at maximizing the pruning ratio, whereas the second relies on a regularization term that imposes a structure on the pruned network topology. We experimentally show that, while the structured approach achieves lower pruning ratio, it yields better end-to-end compression efficiency. We hypothesize that the structured topology of the pruned neural network is the key to higher efficiency of the entropy coder. As a bonus, the network topology is also easier to represent in memory once the network is decompressed, plus inference time is lower as entire operations among tensors are avoided, as we show experimenting on Android devices.

2. BACKGROUND ON NETWORK PRUNING

Parameter pruning builds upon the knowledge that neural networks need to be over-parametrized to be trained on a number

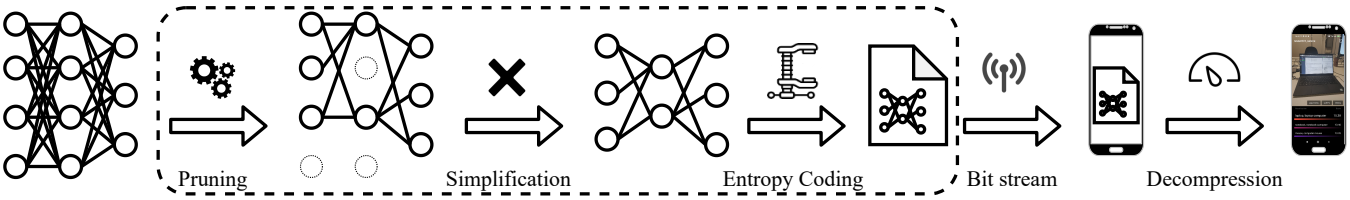


Fig. 1: Our experimental setup; the parts within the dashed box are the object of the MPEG-7 part 17 standard.

of tasks [15, 16]. *One-shot* [9, 17] methods train to completion the full, over-parameterized network, they prune it with some heuristic (e.g. magnitude thresholding) and finally they fine-tune it to recover the accuracy lost due to the pruning stage. A recent work by Frankle and Carbin [9] proposed the *lottery ticket hypothesis*, which is having a large impact on the research community. From a DNN, early in the training, it is possible to extract a sparse sub-network on a one-shot fashion: such sparse network, when trained, can match the accuracy of the original model. *Gradual* pruning methods, instead, train and prune the model simultaneously, spanning a large number of pruning iterations [8, 11, 13, 14]. A recent work compared these two approaches, showing the benefits of iterative pruning in terms of number of parameters of the pruned network [18]. In this work, we target iterative pruning approaches, which can be further divided into two categories.

Unstructured pruning techniques aim at maximizing the pruning ratio, i.e. the number of parameters pruned from the network, regardless of the resulting topology [7, 8]. Dropout-based approaches like *sparse variational dropout*, leverage on a Bayesian interpretation of Gaussian dropout promoting unstructured sparsity [7]. Ullrich *et al.* introduce *soft weight sharing*, through which is possible to introduce redundancy in the network and reduce the amount of stored parameters [6]. The resulting tensors representing the connections between layers are however randomly sparse, i.e. they have no structure. Representing sparse matrices in a memory efficient format is a non-trivial problem, thus high pruning ratios do not necessarily translate into smaller memory footprints.

Structured pruning approaches aim at pruning parameters from the network yet with a constraint on the resulting topology. A recent work [13] proposes a differentiable proxy measure to ℓ_0 norm of parameters, or group of parameters. Minimizing this proxy, it is possible to shrink the total size of the model. The problem with this approach is the significant computational overhead introduced, making it hard to apply to more complex architectures. In [12] a regularizer based on group lasso is proposed, whose goal is to cluster filters in convolutional layers, is proposed. However, such a technique cannot be generalized to bulky fully-connected layers, where most of the complexity (in terms of number of parameters) lies. In the next section we are going to introduce the two state-of-the-art approaches we use within the neural network compression pipeline to obtain unstructured [11] and structured [14] sparsity.

3. PROPOSED COMPRESSION PIPELINE

In this section we describe a neural network compression pipeline as specified in [2] and as illustrated in Fig. 1 (also including a non normative evaluation part).

3.1. Parameter Pruning

During *pruning*, some parameters of the network are removed from the connections graph. We will consider two different pruning strategies.

As an unstructured pruning strategy, we use *LOBSTER* [11]. LOBSTER on the field proves it is a state-of-the-art pruning strategy, achieving higher pruning ratios than other approaches like [5, 7, 8] for challenging tasks like pruning ResNet. It is based on the evaluation at training time of the parameters sensitivity, defined by the variation of the loss function L with respect to the variation of each parameter w_i:

$$S_l(L, w_i) = \left| \frac{\partial L}{\partial w_i} \right|. \tag{1}$$

The term S_l can be exploited as an additional regularizer during training so that parameters having little impact on the loss are pushed towards zero and then pruned to get a sparse network.

As state-of-the-art structured pruning strategy we propose using *SeReNe* [14]. SeReNe generalizes (1) by introducing the concept of neural sensitivity in terms of the variation of the network output $\boldsymbol{y}$ with respect to the variation of the activity p_n of the n-th neuron:

$$S_r(\boldsymbol{y}, \mathcal{W}_n) = \left\| \frac{\partial \boldsymbol{y}}{\partial p_n(\mathcal{W}_n)} \right\|_1. \tag{2}$$

The activity of each neuron, a.k.a. post-synaptic potential, in turns depends on all the neuron's parameters, e.g. a set of weights $\mathcal{W}_n = \{w_{n,i}\}_{i=1}^{P_n}$, being P_n the number of parameters of the n-th neuron. Using a strategy similar to LOBSTER, S_r can be plugged into the regularization term to train a network where the all the parameters $\mathcal{W}_n$ are jointly shrunk. As a result, after pruning the network is not only sparse, but with fewer neurons, e.g. fewer filters for convolutional layers.

Table 1: Experimental results for different network architectures and pruning strategies. Left: percentage of pruned parameters, size of the simplified network topology and size of the compressed bitstream. Right: inference time on different embedded devices: Raspberry Pi 3B (RPi 3B), Huawei P20 (P20), Xiaomi MI 9 (MI9) and Samsung Galaxy S6 lite (S6L).

Dataset	Architecture	Pruning	Pruning ratio [%]	Simplified topology [MB]	Compressed bitstream [MB]	Inference time [ms]			
						RPi 3B	P20	MI9	S6L
CIFAR-10	VGG-16	No pruning	-	60.0	3.6	647	204	153	251
		LOBSTER [11]	**92.44**	58.61	1.61	610	191	146	242
		SeReNe [14]	47.16	**31.02**	**0.34**	**594**	**99**	**85**	**106**
	ResNet-32	No pruning	-	2.0	0.30	580	32	30	31
		LOBSTER [11]	**81.19**	1.96	0.12	545	32	26	30
		SeReNe [14]	52.80	**1.0**	**0.09**	**536**	**25**	**17**	**25**
CIFAR-100	AlexNet	No pruning	-	94.6	10.1	246	131	84	168
		LOBSTER [11]	**98.90**	48.84	0.40	224	95	67	120
		SeReNe [14]	59.87	**37.07**	**0.20**	**186**	**75**	**53**	**96**
ImageNet	ResNet-101	No pruning	-	178.4	26.24	11919	958	416	1008
		LOBSTER [11]	**87.39**	173.87	9.24	11879	956	403	985
		SeReNe [14]	1.09	**172.53**	**7.51**	**11699**	**929**	**371**	**974**

3.2. Topology Simplification

After pruning, the network undergoes *simplification*: in this stage, arcs corresponding to neurons without incoming and/or outgoing connections are removed from the topology. Simplification is mostly effective when the network has been pruned with SeReNe: parameters $\mathcal{W}_n$ can be simply removed, e.g. deleting a whole column in a tensor. See also the discussion about Fig. 2 in Sec. 4.

3.3. Quantization and Coding

Finally, the remaining parameters undergo quantization and entropy coding, yielding a compressed bitstream. We rely on scalar quantization of parameters and DeepCABAC [3, 4] for entropy coding, in accordance to the MPEG-7 part 17 standard. DeepCABAC represents an evolution of the context adaptive binary arithmetic coding used in video that includes model quantization, binarization and arithmetic coding.[1]

4. EXPERIMENTAL RESULTS

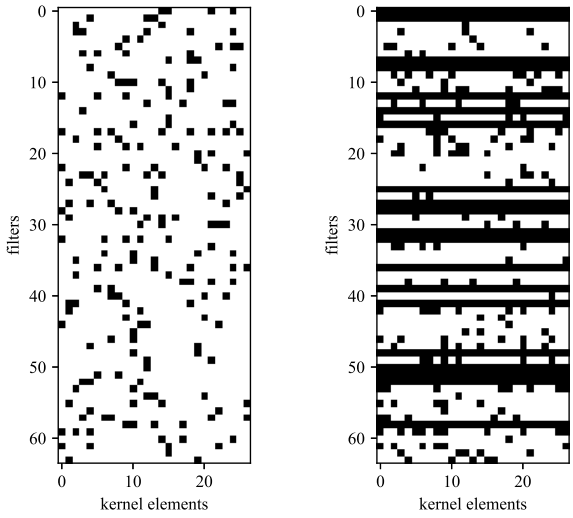
Tab. 1 shows the results of our experiments with the popular VGG-16, ResNet and AlexNet architectures for image classification over the CIFAR-10, CIFAR-100 and ImageNet datasets. All the architectures are compressed according to the scheme described in the previous section, i.e. we alternatively prune the networks with LOBSTER [11](unstructured) and SeReNe [14] (structured). The size of the simplified networks refers to the case where the parameters are represented over 32-bit floating-point values, compliant to the IEEE 754 standard. All results are obtained with fixed quantization step equal to 2^{-15}. As expected, LOBSTER yields the highest compression ratio, i.e. removes more parameters from the network, whereas SeReNe yields more compact simplified topologies. Of course, the more general the task, the least the parameters/neurons to be removed without performance loss (for example, on ImageNet the parameters removed in proportion are less than for other architectures).

Fig. 2 shows the parameter pruning map for the first convolutional layer of VGG-16 trained over CIFAR-10 (64 convolutional neurons, 3 filters sized 3×3 per neuron) for both LOBSTER (Fig. 2a) and SeReNe (Fig. 2b). Here, black lines represent neurons for which all parameters were pruned by SeReNe, and thus are not represented altogether in the simplified topology, yielding reduced size simplified networks. As a result, structured pruning always yields the best end-to-end compression efficiency in terms of size of compressed bitstream size. In particular, for VGG-16 trained on CIFAR-10, the SeReNE-pruned network bitstream is about 5 times smaller than the LOBSTER reference.

Clearly, to guarantee optimal encoding into the final bitstream, one needs not only to reduce the number of parameters in the network, but also that their entropy is low. Our experiments highlight that SeReNe is very effective also in terms of parameters entropy or compressibility. To spot this effect let us observe the compression ratio C_p achieved by DeepCABAC after simplification, measured as the ratio between the final compressed bitstream size and the simplified floating-point network: in the VGG-16 case we get $C_p = 0.0275$ for unstructured and $C_p = 0.011$ for structured pruning: the parameters retained with SeReNe can be compressed about 3 times more.

To summarize, unstructured pruning is effective in setting the largest possible amount of parameters to zeros, but the

[1] https://github.com/fraunhoferhhi/DeepCABAC



(a) (b)

Fig. 2: Representation of the first convolutional layer for VGG-16 trained on CIFAR-10: unstructured pruning LOBSTER (a) and structured pruning SeReNe (b). Black pixels are pruned parameters.

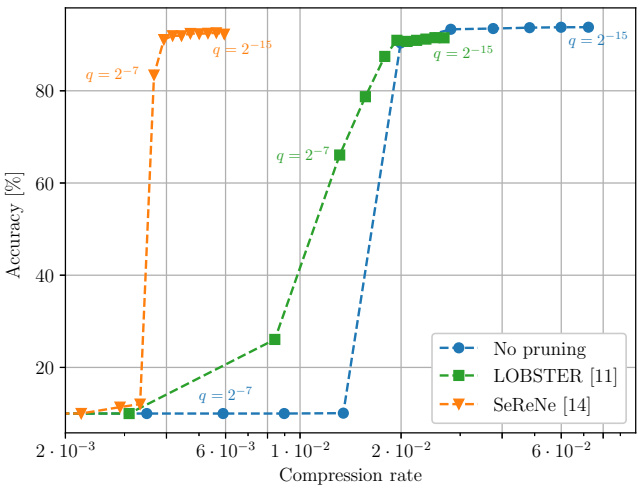


Fig. 3: Classification accuracy vs. compression rate for VGG-16 on CIFAR-10 for different pruning strategies. The compression rate is the ratio between the model's size after DeepCABAC and the original size. Lower compression rate means better compression efficiency.

corresponding network tensors are sparse, and the remaining parameters are costly to represent. On the other hand, structured pruning provides a counter-intuitive behavior since one gets more non-zeros parameters, but their structure can be exploited for better compression. Not only the tensor can be trivially simplified (e.g. by completely dropping some dimensions), but the entropy of the symbols to be encoded with DeepCABAC turns to be lower. This result is likely due to the favourable interplay between the neural sensitivity regularizer (2) and the context modeling in DeepCABAC: indeed, (2) amounts at imposing a constraint onto a set of weights $\mathcal{W}_n$ in the context of a given neuron; the same regularity can be exploited by DeepCABAC to jointly represent the same context. On the contrary, unstructured regularization (1) behaves independently on every weight, with lower chances to create homogeneous contexts for the following entropy coder.

Finally, Fig. 3 shows the compression-accuracy trade-off for VGG-16 over CIFAR-10 dataset for different DeepCABAC quantization steps $q \in \left[2^{-15}; 1\right]$ range. SeReNe's structured pruning yields comparable accuracy at far lower encoded bitstream rate, outperforming by a large margin LOBSTER.

4.1. Inference Time

Tab. 1 also shows the inference time (averaged on 1k trials) when the decompressed simplified architecture is deployed on embedded devices.The experiments have been worked on the following devices:

- RaspBerry Pi 3B (RPi 3B): Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB RAM;
- Huawei P20 phone (P20): 4x2.36 GHz Cortex-A73 + 4x1.84 GHz Cortex-A53 processors, 4GB RAM;
- Xiaomi MI 9 phone (MI9): 1x2.84 GHz Kyro 485 + 3x2.42 GHz Kyro 485 + 4x1.80 GHz Kyro 485, 6GB RAM;
- Samsung Galaxy S6 lite tablet (S6L): 4x2.3 GHz Cortex-A73 + 4x1.7 GHz Cortex-A53, 4GB RAM.

Structured sparsity always yield lower inference time as a side benefit of the more compact representation of the network into the device memory and the fewer matrix-vector multiplication required at inference time.

5. CONCLUSIONS

In this work we evaluated the role of unstructured and structured parameter pruning approaches in a standardized neural network compression pipeline. The surprising result of our experiments is that structured pruning enables better end-to-end compression despite lower pruning ratios. We explain this finding in part with the structure of the pruned network that is easier to simplify dropping entire parts of the tensors and in part with the lower entropy of the residual symbols processed by the DeepCABAC encoder. Future works include the design of a entropy-aware regularization strategy in order to minimize at training/pruning time the model's size after compression.

6. REFERENCES

[1] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[2] The Motion Picture Expert Group, "ISO/IEC DIS 15938-17 Compression of neural networks for multimedia content description and analysis," 2021.

[3] David Neumann, Felix Sattler, Heiner Kirchhoffer, Simon Wiedemann, Karsten Müller, Heiko Schwarz, Thomas Wiegand, Detlev Marpe, and Wojciech Samek, "Deepcabac: Plug & play compression of neural network weights and weight updates," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 21–25.

[4] Simon Wiedemann, Heiner Kirchhoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinč, David Neumann, Tung Nguyen, Heiko Schwarz, Thomas Wiegand, et al., "Deepcabac: A universal compression algorithm for deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 700–714, 2020.

[5] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[6] K. Ullrich, M. Welling, and E. Meeds, "Soft weight-sharing for neural network compression," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019, cited By 2.

[7] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *34th International Conference on Machine Learning, ICML 2017*, 2017, vol. 5, pp. 3854–3863, cited By 29.

[8] Enzo Tartaglione, Skjalg Lepsøy, Attilio Fiandrotti, and Gianluca Francini, "Learning sparse neural networks via sensitivity-driven regularization," in *Advances in neural information processing systems*, 2018, pp. 3878–3888.

[9] Jonathan Frankle and Michael Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[10] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[11] Enzo Tartaglione, Andrea Bragagnolo, Attilio Fiandrotti, and Marco Grangetto, "Loss-based sensitivity regularization: towards deep sparse neural networks," *arXiv preprint arXiv:2011.09905*, 2020.

[12] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2074–2082.

[13] Christos Louizos, Max Welling, and Diederik P Kingma, "Learning sparse neural networks through l_0 regularization," *International Conference on Learning Representations, ICLR 2018*, 2018.

[14] Enzo Tartaglione, Andrea Bragagnolo, Francesco Odierna, Attilio Fiandrotti, and Marco Grangetto, "Serene: Sensitivity-based regularization of neurons for structured sparsity in neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2021.

[15] Hrushikesh N Mhaskar and Tomaso Poggio, "Deep vs. shallow networks: An approximation theory perspective," *Analysis and Applications*, vol. 14, no. 06, pp. 829–848, 2016.

[16] A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz, "Sgd learns over-parameterized networks that provably generalize on linearly separable data," 2018.

[17] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[18] Enzo Tartaglione, Andrea Bragagnolo, and Marco Grangetto, "Pruning artificial neural networks: A way to find well-generalizing, high-entropy sharp minima," in *Artificial Neural Networks and Machine Learning – ICANN 2020*, Igor Farkaš, Paolo Masulli, and Stefan Wermter, Eds., Cham, 2020, pp. 67–78, Springer International Publishing.